Zeitschrift: Elemente der Mathematik
Herausgeber: Schweizerische Mathematische Gesellschaft
Band: 22 (1967)
Heft: 4

Artikel: Tangram : ein Puzzle-Problem für den Computer
Autor: Läuchli, P.
DOI: https://doi.org/10.5169/seals-25359

ELEMENTE DER MATHEMATIK
Revue de mathématiques élémentaires – Rivista di matematica elementare
Zeitschrift zur Pflege der Mathematik
und zur Förderung des mathematisch-physikalischen Unterrichts
Publiziert mit Unterstützung des Schweizerischen Nationalfonds
zur Förderung der wissenschaftlichen Forschung

El. Math. Band XXII Heft 4 Seiten 73–96 10. Juli 1967

Tangram – ein Puzzle-Problem für den Computer

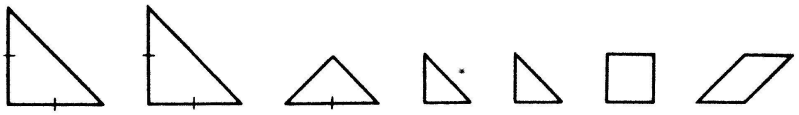
1 Die Frage, auf wieviele Arten sich ein Paket von n Karten auf m Spieler verteilen lasse, kann mit elementaren kombinatorischen Mitteln durch einen geschlossenen Ausdruck in n und m beantwortet werden.

Das klassische Geldwechselproblem (auf wieviele Arten lässt sich eine Summe von n Rappen in den existierenden Münzsorten wechseln?) ist, obschon um einiges schwieriger als die erste Frage, einer analytischen Behandlung in dem Sinne noch zugänglich, dass z. B. die erzeugende Funktion explizit angegeben werden kann, deren Reihenentwicklungskoeffizienten gerade die gesuchten Anzahlen sind (siehe [2][1])).

Bei noch komplizierteren kombinatorischen Aufgaben ist aber bald einmal der Punkt erreicht, wo die Lösung nur noch durch effektives Abzählen aller möglichen Fälle erhalten wird. Allerdings – und dies sollen die folgenden Ausführungen illustrieren – kann der Abzählprozess mehr oder weniger geschickt angelegt werden. Jedenfalls nimmt bei verwickelteren Problemen der Aufwand derartige Ausmasse an, dass man nicht mehr ohne die Hilfe eines Computers durchkommt. Damit tauchen aber sofort Probleme der Formulierung und Darstellung auf, welche beim Arbeiten mit Papier und Bleistift allein weit weniger ins Gewicht fallen.

Die Anregung zu den folgenden Betrachtungen verdanke ich meinem Kollegen Professor E. SPECKER.

2 Unter dem Stichwort *Tangram* findet man im Oxford English Dictionary die folgenden Erklärungen: *The name given to a Chinese geometrical puzzle consisting of a square dissected into five triangles, a square, and a rhomboid, which can be combined so as to make two equal squares, and also so as to form several hundred figures, having a rude resemblance to houses, boats, bottles, glasses, urns, birds, beasts, men, etc.*



Figur 1
Die Steine des Tangram-Puzzles.

[1]) Die Ziffern in eckigen Klammern verweisen auf das Literaturverzeichnis, Seite 85.

Zunächst stellt sich also die Aufgabe, die sieben in Figur 1 wiedergegebenen Steine zu einem Quadrat, bzw. zu den zwei kleineren Quadraten zusammenzusetzen. Allerdings sind diese beiden Aufgaben doch etwas zu dürftig, als dass es sich lohnen würde, dafür ein Computer-Programm zu schreiben. Auf der anderen Seite geht die Frage nach allen möglichen Zusammensetzungen wieder zu weit, da offensichtlich überabzählbar viele verschiedene Figuren aus den gegebenen Steinen gebildet werden können. Und diejenigen Figuren herauszusondern, welche die erwähnte Ähnlichkeit mit Häusern usw. zeigen, dürfte die Fähigkeiten unserer heutigen Automaten noch übersteigen.

Eine willkürliche, aber gewiss sinnvolle Beschränkung ergibt sich nun aus der Forderung, dass die Figuren *konvex* sein sollen. (Man möge sich schon an dieser Stelle überlegen, ob nur endlich viele Lösungen existieren und, wenn ja, die Anzahl grob zu schätzen versuchen).

Für die folgenden drei Probleme wurden im Laufe der Zeit vom Verfasser Lösungsverfahren programmiert:

Programm 1: Gegeben eine konvexe Figur mit dem richtigen Flächeninhalt. Kann die Figur aus den Tangram-Steinen aufgebaut werden?

Programm 2: Konstruktion aller verschiedenen konvexen Zusammensetzungen, die sich aus den Tangram-Steinen bilden lassen, gruppiert nach dem äusseren Umriss.

Programm 3: Verallgemeinerung von Programm 2 für eine beliebige Anzahl von (im weiter unten präzisierten Sinne) beliebigen Steinen.

Unsere Puzzle-Programme, und zwar vor allem Programm 3, sollen den Anlass darstellen, um im Rahmen dieser Zeitschrift einige typische Schwierigkeiten und Fragen zu diskutieren, die sich bei der Programmierung derartiger nichtnumerischer Aufgaben immer einstellen. Solche Fragen prinzipieller Natur sind zum Beispiel die folgenden:

1. Wie lässt sich ein Sachverhalt, der mir klar ist, und den ich in Worten beschreiben kann – besonders gut anhand einer Skizze! – maschinengerecht formulieren? (Unsere digitalen Computer sind daraufhin konzipiert, mit Zahlen zu rechnen und nicht, mit geometrischen Figuren umzugehen).

2. Sollen in das Programm viele raffinierte Überlegungen gesteckt werden, womit unter Umständen die Rechenzeit beträchtlich abgekürzt werden kann und dafür das Programm lang und kompliziert wird, oder soll das Programm einfach und übersichtlich angelegt sein und (zum Beispiel bei kombinatorischen Aufgaben) viele Fälle stur durchprobieren, welche bei näherer Betrachtung zum vornherein hätten ausgeschlossen werden können?

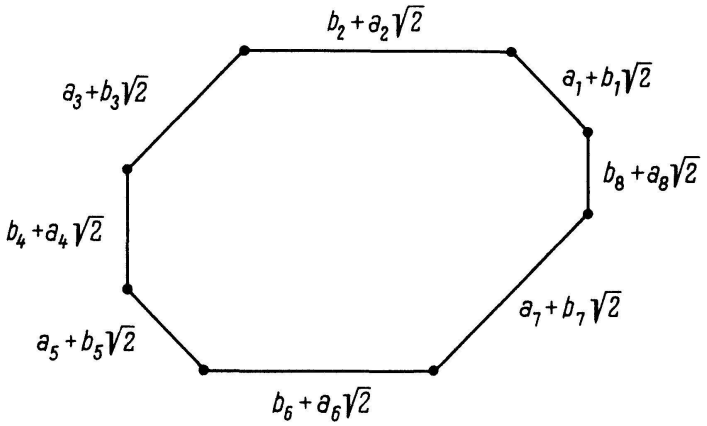
3. Wie zuverlässig sind die Resultate, welche mein Programm liefert? (Die Lösungen eines linearen Gleichungssystems sind relativ leicht auf ihre Richtigkeit zu prüfen. Dies gilt keineswegs für Probleme, wie das hier vorliegende.)

3 Im folgenden wird nun in groben Zügen geschildert, was sich abspielt vom Moment an, da das Programm von Lochkarten die gegebene Anzahl und Form der Puzzle-Steine liest, bis zur automatischen Ausgabe der konstruierten Figuren, zum Beispiel mittels eines am Computer angeschlossenen Zeichengerätes (siehe Figur 8). Bevor aber auf das Konstruktionsverfahren eingegangen werden kann, muss schon etwas über die maschinenmässige Erfassung der vorkommenden Figuren gesagt werden, da so erst die Aufgabenstellung präzise formuliert werden kann: Die spezielle

Gestalt der in Figur 1 dargestellten Tangram-Steine legt nämlich die Vermutung nahe, dass sich alle aus ihnen zusammengesetzten Figuren in ein Quadratnetz legen lassen, wodurch die ziffernmässige Verschlüsselung einer solchen Figur auf einfachste Weise durch die Angabe von besetzten Feldnummern gegeben werden könnte. Diese Vermutung trifft aber nur für *konvexe* Figuren zu; es gilt dann der etwas allgemeinere Satz: Ein konvexes Polygon, das in kongruente gleichschenklig rechtwinklige Dreiecke (Elementardreiecke genannt) zerlegt werden kann, lässt sich so in ein Quadratnetz legen, dass die Ecken aller Elementardreiecke auf Netzpunkte fallen.

Aus dem Satz folgt sofort, dass unser Problem nur endlich viele Lösungen besitzt; für jede Gesamtfläche F lässt sich diese Anzahl leicht einschränken. Die Aussage des Satzes scheint durchaus plausibel, muss aber doch genau geprüft werden, da die Voraussetzung der Konvexität auf nicht ganz durchsichtige Art hineinspielt.

Knappe Skizze des Beweises: Aus der Voraussetzung folgt, dass das Polygon nur Winkel von $k \cdot 45°$ $(k = 1, 2, 3)$ besitzt, also im allgemeinsten Falle ein Achteck gemäss Figur 2 ist. Die a_i und b_i sind nichtnegative ganze Zahlen, wobei die Katheten des Elementardreiecks gleich 1 angenommen werden. Es ist nun weiter zu zeigen, dass, falls (für $F > 0$) nicht alle $a_i = 0$ oder alle $b_i = 0$ sind, im Ausdruck für F $\sqrt{2}$-Anteile stehen bleiben, im Widerspruch zur Voraussetzung. Das ist nicht ohne weiteres ein-



Figur 2
Allgemeinste Gestalt des konvexen Netzpolygons.

zusehen, wenn F als Fläche eines Rechtecks mit abgeschnittenen Ecken berechnet wird, da dann im quadratischen Polynom für F auch negative Koeffizienten auftreten. Wenn hingegen das Polygon durch die von einer Ecke ausgehenden Diagonalen in Dreiecke zerlegt wird, erscheinen nur positive Koeffizienten, und die Zwischenbehauptung lässt sich leicht verifizieren.

Somit kann der Umriss des Polygons in das Quadratnetz gelegt werden, und die ganze Figur lässt sich in der Folge von aussen her durch Elementardreiecke, die ebenfalls im Netz liegen, abbauen. Damit ist der Satz bewiesen.

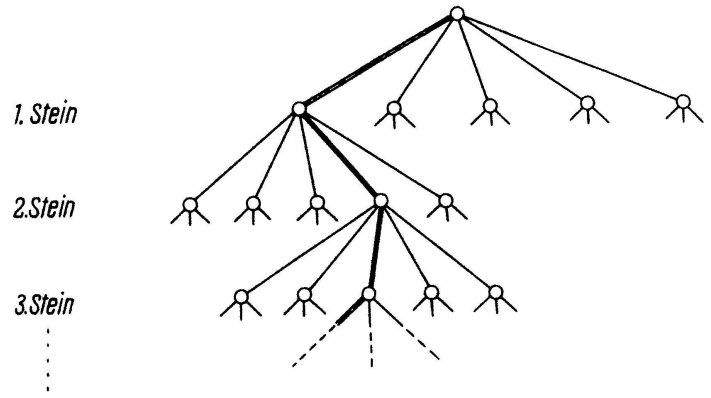
Die Aufgabenstellung für das vorher erwähnte Programm 3 soll nun dahingehend präzisiert werden, dass als Steine nur solche Polygone zugelassen werden, die sich im Sinne des obigen Satzes in das Quadratnetz legen lassen. Die Einzelsteine brauchen aber selbst durchaus nicht konvex zu sein.

4 Um ein Konstruktionsverfahren aufzustellen, darf man sich ruhig einmal überlegen, auf welche Weise man wirkliche Steine von Hand auf dem Tisch auslegen würde:

Sicher würde man in irgendeiner Reihenfolge die Steine aneinanderfügen und so versuchen, eine konvexe Figur zu erzeugen. Vielleicht würde man gelegentlich schon vor dem letzten Stein feststellen, dass keine Aussicht besteht, die Teilkonfiguration zu einer konvexen Figur zu ergänzen und würde dann schon den betreffenden Stein verschieben. Wenn man endlich eine Lösung gefunden hätte, würde man wohl nicht völlig neu beginnen, sondern zuerst versuchen, den letzten Stein anders zu plazieren, wenn nötig noch weitere Steine usw.

Wenn es nun darum geht, ein derartiges Verfahren zu programmieren, so sieht man zunächst einmal, dass kein Grund mehr dafür besteht, die Steine so aneinander zu legen, dass sie sich von Anfang an berühren. Im Gegenteil, wenn die Steine in einer festen Reihenfolge gesetzt werden sollen, was sicher Vorteile bietet, ist dies gar nicht mehr möglich. Wir werden also innerhalb eines maximalen Bereiches die Steine der Reihe nach setzen, wobei im gegebenen Bereich alle Lagen zulässig sind, ausgenommen natürlich diejenigen, bei welchen mit schon früher gesetzten Steinen Kollisionen entstünden.

Betrachtet man den inneren Zusammenhang der einzelnen Schritte im eben angedeuteten Verfahren, so erkennt man leicht die Struktur eines *Baumes*, der sich an allen Zwischenstellen stark verzweigt: Jeder Punkt im baumartigen Graphen entspricht dem Setzen eines Steines, die von diesem Punkte ausgehenden Zweige führen zu allen möglichen Positionen des nächsten Steines. Und das Konstruktionsverfahren besteht nun darin, dass, ausgehend vom obersten Punkt in der schematischen Darstellung von Figur 3, der ganze Baum systematisch nach solchen Endpunkten abgesucht wird, welche einer Lösung entsprechen. Dabei geht man nach einer festen Ordnung vor, indem man zum Beispiel in jedem Punkt den am meisten links liegenden noch unbenützten Weg wählt. Der Prozess als ganzes bietet das Bild einer auf- und abwärts pulsierenden Bewegung des laufenden Punktes im Graphen.



Figur 3

Baumstruktur, schematisch dargestellt. Eine momentane Situation im Konstruktionsverfahren.

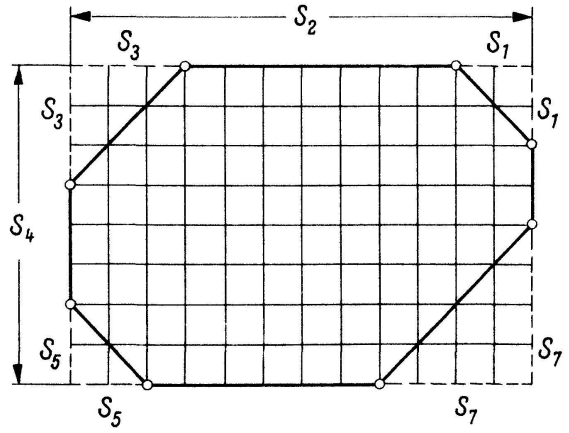
Da die Anzahl der Endpunkte des Baumes, d.h. also die Anzahl der Kandidaten für eine Lösung mit der Zahl der Stufen (Puzzle: Zahl der Steine) exponentiell anwächst, erhebt sich bald sehr gebieterisch die Forderung, den Suchprozess derart zu verfeinern, dass «unfruchtbare» Zweige schon möglich früh, d.h. weit oben im Graphen erkannt und damit eliminiert werden können. Zur Illustration dieser Forderung diene eine kleine Überschlagsrechnung für den Fall Tangram: Man überlegt sich

leicht, dass alle konvexen Netzfiguren mit der Fläche $F = 16$ Elementardreiecke in einem Quadrat der Seitenlänge 9 Platz finden. Wenn man weiter berücksichtigt, dass, ausser dem quadratischen, alle Steine in vier verschiedenen Orientierungen gesetzt werden können, ergibt sich eine Anzahl von ca. $5 \cdot 10^{16}$ Endpunkten des Baumes. Mit einem Computer, der 10^3 Fälle pro Sekunde erledigt, käme man so auf eine Rechenzeit von einer Million Jahren.

Um hier Abhilfe zu schaffen, wurde folgendes versucht: In Programm 2 wird jedesmal, wenn ein neuer Stein gesetzt ist, bei dem keine Überdeckung mit schon besetzten Feldern stattfindet, das kleinste konvexe Netzachteck bestimmt, welches alle jetzt besetzten Felder enthält (also eine Art «diskrete konvexe Hülle»). Wenn nun diese Hülle eine Fläche $F > 16$ erreicht, scheidet die Lage des neuen Steines aus; ist $F \leqslant 16$, so kann der Stein bleiben, aber bei $F = 16$ werden alle folgenden Steine nur noch in die Hülle hineingelegt, bei $F < 16$ sind sie zunächst noch frei.

Trotz diesen Massnahmen muss in diesem Programm bis die letzte Lösung gefunden ist immer noch 154396mal ein Stein gesetzt werden (nur die Fälle ohne Kollision gezählt), und die Rechenzeit ist entsprechend lang. Das geschilderte Verfahren hat den weiteren Nachteil, dass die Lösungen in regelloser Folge entdeckt werden und somit alle gespeichert und am Schluss nach dem äusseren Umriss der Figur sortiert werden müssen.

In Programm 3 wird ein anderer Weg beschritten: Wir bestimmen zunächst in einem Vorspann alle konvexen Netzachtecke der (durch die Steine festgelegten) Fläche F, d.h. alle möglichen Umrisse. Die Gestalt eines Umrisses ist eindeutig definiert zum Beispiel durch die Grössen s_2, s_4, s_1, s_3, s_5, s_7 (siehe Figur 4). Dagegen kommen teilweise für verschiedene Sixtupel $s_2, \ldots$ kongruente Achtecke heraus.



Figur 4
Festlegung der Umrisse.

Um die Aufgabe genau zu formulieren, betrachten wir die ganzzahligen nichtnegativen Lösungen der Gleichung

$$F = 2\, s_2\, s_4 - (s_1^2 + s_3^2 + s_5^2 + s_7^2)$$

(F gegeben, immer gemessen in Elementardreiecken), welche den Nebenbedingungen

$$s_1 + s_3 \leq s_2; \qquad s_3 + s_5 \leq s_4$$
$$s_5 + s_7 \leq s_2; \qquad s_1 + s_7 \leq s_4$$

genügen. Diese Menge von Lösungen zerfällt in Klassen (Transitivitätsgebiete), innerhalb derer die Lösungen infolge der Symmetrietransformationen des Quadrates permutiert werden. Aus jeder solchen Klasse soll ein Repräsentant angegeben werden.

Die Anzahl N dieser Klassen (d.h. die Anzahl verschiedener Achtecke) ist offenbar eine sehr komplizierte zahlentheoretische Funktion von F, sie verläuft zum Beispiel keineswegs monoton. Einige Werte:

F	1	2	3	4	5	6	7	8	9	10	16 (Tangram)
N	1	3	2	6	3	7	5	11	5	10	20

Das Teilproblem, die im obigen Sinne wesentlich verschiedenen Lösungen der angegebenen diophantischen Gleichung einigermassen rationell zu erzeugen, stellt übrigens eine sehr dankbare Übungsaufgabe im Programmieren dar.

Im Hauptprogramm werden nun die Steine nur noch in allen möglichen Positionen in die vorbestimmten Umrisse hineingelegt. Damit ist der Baum gegenüber Programm 2 ganz beträchtlich zusammengestutzt worden. Im Tangrambeispiel, nach Programm 3 durchgerechnet, muss nur noch 13616mal ein Stein gesetzt werden.

5 Ein weiterer Aspekt kommt nun noch dazu, der das Verfahren wesentlich kompliziert, dafür den Baum nochmals erheblich zu reduzieren gestattet: Wenn, wie bis dahin unterstellt, jeder Stein an jede freie Stelle gelegt wird, dann erzeugt man in manchen Fällen dieselbe Figur mehrmals, nur eventuell in anderer Lage. Und zwar tritt dieser Fall genau dann ein, wenn der Umriss Symmetrien besitzt, oder wenn mindestens ein Stein in mehreren Exemplaren vorhanden ist. Es ist nun äusserst wichtig, dass Ansätze zu solchen überflüssigen Konfigurationen schon im Keime erstickt, d.h. möglichst hoch oben im Baume entdeckt werden, und nicht erst, wenn alle Steine gesetzt sind. Damit können wieder ganze Zweige eliminiert werden.

Ein möglicher Weg, dieses Ziel zu erreichen, ist der folgende: Die Positionen der einzelnen Steine müssen ohnehin durchnumeriert sein, da das Durchlaufen des Baumes nach einer festen Ordnung zu erfolgen hat. (Zur Definition der «Position» eines Steines gehören sowohl die Lage eines starr mit ihm verbundenen Bezugspunktes als auch seine Orientierung. Je nach Symmetrieeigenschaft besitzt ein Stein eine bis acht verschiedene Orientierungen). Damit ergibt sich eine lexikographische Ordnung innerhalb aller Teilkonfigurationen von der gleichen Anzahl Steine, indem man dem zuerst gesetzten Stein das höchste Gewicht zumisst usw. Nun wird jedesmal, wenn beim Abwärtsschreiten im Baum eine neue Steinsorte an die Reihe kommt (wir setzen voraus, dass alle Exemplare desselben Steines direkt aufeinanderfolgend gesetzt werden), die Symmetriegruppe U der momentanen Teilkonfiguration – d.h. bei der ersten Steinsorte diejenige des äusseren Umrisses – festgestellt. (Mit zunehmender Tiefe im Baum gehen höchstens Symmetrien verloren, es können keine neuen hinzukommen.) Wenn jetzt innerhalb dieser Steinsorte ein neues Exemplar an der Reihe ist, wird es erstens nur in Positionen gebracht, welche in der Numerierung nach den schon gesetzten Steinen derselben Sorte kommen. Damit vermeiden wir die kongruenten Konfigurationen, welche durch Permutation von gleichen Steinen ineinander übergehen. Wenn zweitens das neue Exemplar wirklich im noch freien Bereich gesetzt werden kann, wird diese Teilkonfiguration versuchsweise in alle Lagen gemäss U gebracht. Falls eine dieser Lagen in der oben erwähnten lexikographischen Ordnung früher kommt als die direkt erzeugte Konfiguration, dann scheidet diese sofort aus,

da sie nur auf Lösungen führen könnte, welche kongruent zu vorher erzeugten wären.

Um nochmals das Beispiel Tangram heranzuziehen: 13616mal wird versucht, einen Stein zu setzen. Dabei hat der Stein in 1451 Fällen wirklich Platz in der Restfigur, und in 1326 von diesen Fällen erlaubt auch der Symmetrie-Test diese Position.

Der relativ kleine Unterschied zwischen den letzten zwei Zahlen ist vielleicht etwas irreführend; man möge jedoch bedenken, dass jedemal, wenn der Symmetrie-Test eine Position ausschliesst, dadurch ein ganzer Zweig aus dem Baum geschnitten wird, welcher sonst auch durchlaufen würde.

6 In diesem Abschnitt sollen einige besonders wichtige Details des Programmes in zwangloser Folge etwas näher erläutert werden.

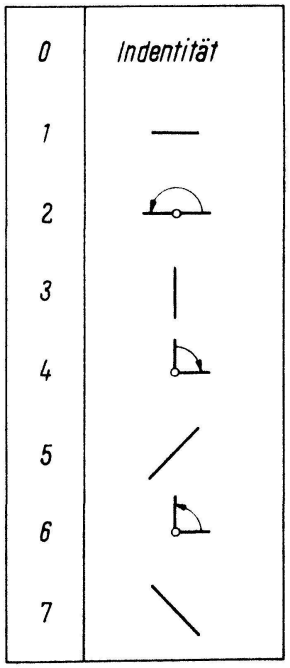
Für die *Darstellung*, d.h. ziffernmässige Verschlüsselung von Einzelsteinen und ganzen Konfigurationen werden zwei verschiedene Prinzipien nebeneinander verwendet:

1. Liste der besetzten Felder im Quadratnetz. Für jedes Feld benötigt man dabei die Koordinaten zum Beispiel seines Zentrums, sowie die Angabe, ob das Feld ganz oder auf welche der vier möglichen Arten nur durch ein Elementardreieck besetzt ist.

2. Liste der Kanten (auch innere) der Figur. Kanten definiert durch die Koordinaten der beiden Endpunkte.

So liest zum Beispiel zu Beginn das Programm die Puzzle-Steine in der Kantendarstellung von Karten (ganzzahlige Koordinaten, Quadratnetz), da diese Form für die Eingabe bequemer ist; es wandelt die Information aber auch noch in die Felderdarstellung um. Nachher werden beide Darstellungen verwendet: Für die Angabe des momentan noch freien Teiles eines Achteckes kommt nur die Felderdarstellung in Frage, denn dieses Gebiet kann eine sehr komplizierte Form besitzen; es braucht zum Beispiel nicht zusammenhängend zu sein. Auch die Prüfung, ob ein probeweise gesetzter Stein in diesem Gebiete Platz findet, lässt sich so durch Vergleich von Feldnummern relativ leicht durchführen. Wenn dagegen ein Stein in einer bestimmten Position diesen, sowie auch den Symmetrie-Test bestanden hat, dann werden seine Kanten entsprechend dieser Lage gespeichert, da später bei der Ausgabe einer Lösungsfigur die Kanten «gezeichnet» werden sollen.

Bei der *Eingabe der Steine* am Anfang wird natürlich nur eine Lage für jeden Stein verlangt; das Programm muss selbst herausfinden, welche Symmetrien vorhanden sind. Dies

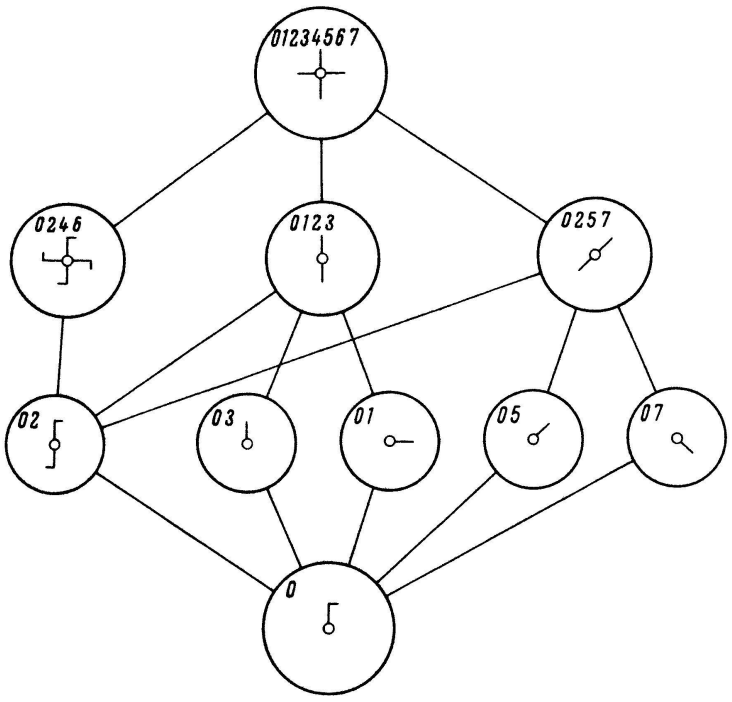


Figur 5

Die Elemente der Drehspiegelungsgruppe des Quadrates.

geschieht so, dass zuerst das umbeschriebene Achteck bestimmt wird (es sei nochmals daran erinnert, dass die Einzelsteine nicht konvex vorausgesetzt werden). Eine allfällige Symmetrie des Steines muss auch das Achteck besitzen. Also entscheidet man zuerst mit einem einfachen Unterprogramm, das später auch wieder für die Umrisse verwendet wird, auf Grund der Grössen s_2, s_4, s_1, s_3, s_5, s_7 welche der Symmetrien 1 bis 7 (siehe Figur 5) beim Achteck vorhanden sind, und dann für diese, ob sie auch beim einbeschriebenen Puzzle-Stein gelten. Natürlich macht man davon Gebrauch, dass eine Symmetrie-Achse durch das «Zentrum» des Achtecks gehen, bzw. ein Symmetrie-Zentrum mit ihm zusammenfallen muss.

Sind so die Symmetrien ausgemacht worden, dann braucht es für die Bestimmung des Symmetrie-Typs nur noch eine recht einfache Entschlüsselung. Es sei dem Leser überlassen, auf Grund des in Figur 6 symbolisch dargestellten Untergruppenverbandes der Quadratgruppe selbst eine derartige Entschlüsselungspyramide zu entwerfen.



Figur 6

Die 10 Untergruppen der Quadratgruppe.
Die Ziffern in den Kreisen entsprechen der Numerierung der Gruppenelemente in Figur 5.

Später, bei der Durchführung des bereits erwähnten *Symmetrie-Tests*, welcher zur Vermeidung kongruenter Konfigurationen nötig ist, wird von den Symmetrie-Eigenschaften der Steine Gebrauch gemacht. Es muss dann festgestellt werden können, in welche Orientierung ein Stein gerät, der einem bestimmten Symmetrie-Typus angehört, wenn auf ihn in einer bestimmten Orientierung eine bestimmte Transformation ausgeübt wird. Zu diesem Zweck bereitet das Programm am Anfang eine Tabelle vor, welche die benötigte Information enthält. Man beachte, dass «Orientierung» eines Steines hier gleichbedeutend ist mit Linksrestklasse modulo seiner Symmetriegruppe.

Es sei nur am Rande vermerkt, dass beim Symmetrie-Test durch einen zusätzlichen Kunstgriff dafür gesorgt wird, dass im allgemeinen nicht alle Exemplare der laufenden Steinsorte den Symmetrie-Transformationen unterworfen werden müssen, was sich zeitsparend auswirkt.

Eine wichtige Frage, welche bei jedem Programm sehr sorgfältig studiert werden muss, betrifft den *Umfang der zu speichernden Information*. Oft gilt es, Speicherbedarf und Rechenzeit gegeneinander abzuwägen. Ein Beispiel aus unserem Puzzle-Programm möge dies illustrieren: Die Liste der momentan noch unbesetzten Felder innerhalb des Umrisses

muss jederzeit zur Verfügung stehen. Sie wird sich aber, entsprechend der Auf- und Abbewegung im Baum, ständig verlängern und verkürzen. Nun stellen sich zwei Möglichkeiten für die Behandlung dieser Liste: Entweder speichert man bei jedem Abwärtsschritt im Baum, d.h. für jedes Niveau (nicht für jeden Punkt!) die Liste ab und findet sie somit wieder beim Aufwärtsschreiten. Oder man führt die Liste nur in einem Exemplar und muss dann bei jedem Aufwärtsschritt den früher an dieser Stelle gesetzten Stein nochmals suchen und die Liste um dessen Felder verlängern. Offensichtlich benötigt die zweite Variante weniger Speicherplatz, aber mehr Zeit. Da in unserem Programm vor allem die Rechenzeit forciert werden sollte, haben wir die erste Variante gewählt.

7 Bevor auf einzelne Resultate näher eingegangen wird, soll erwähnt werden, dass die hier skizzierten Programme in der Formelsprache ALGOL geschrieben wurden. Diese Sprache erlaubt es, auch derartige nichtnumerische Probleme in einer sehr handlichen und leicht lesbaren Form zu programmieren, soweit nicht (z.B. um Speicherplatz oder Zeit zu sparen) von der maschineninternen Darstellung der Zahlen im Dualsystem explizit Gebrauch gemacht werden muss. Allerdings werden die Möglichkeiten der Sprache in unserem Falle nicht ausgeschöpft, da wir nur ganzzahlig rechnen. Das Programm 3, von welchem jetzt immer die Rede war, hat eine Länge von ca. 850 Zeilen (Karten).

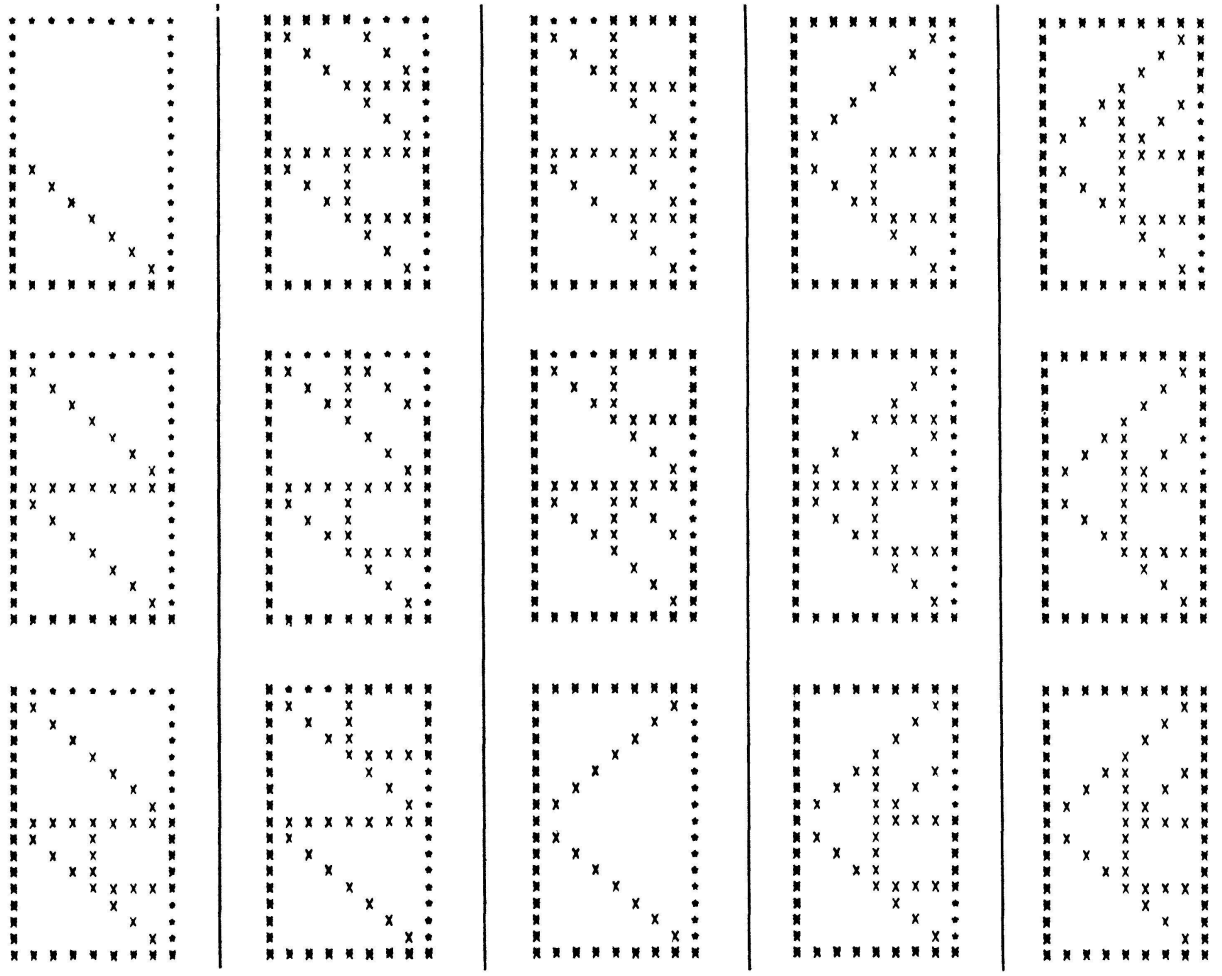
Einige Angaben zum Beispiel Tangram, von welchem unsere Betrachtungen ja ausgegangen sind: Zur Fläche F = 16 (Elementardreiecke) gibt es 20 verschiedene konvexe Achtecke. Von diesen lassen sich 13 in die Tangram-Steine zerlegen. Bei den übrigen 7 Umrissen ist dies zum Teil trivialerweise nicht möglich, da schon das grosse Dreieck nicht Platz hat (z.B. Rechteck mit den Seiten 1 und 8), in anderen Fällen muss der Baum lange abgesucht werden bis herauskommt, dass keine Lösung existiert (z.B. Parallelogramm mit den Seiten 2 und $4\sqrt{2}$. 672 zum Teil erfolgreiche Versuche, einen Stein zu setzen). Selbstverständlich übt die Reihenfolge, in welcher die verschiedenen Steine dem Programm gegeben und in welcher sie dann auch gesetzt werden, einen wesentlichen Einfluss aus auf die notwendige Anzahl von Versuchen; es wäre zum Beispiel ungünstiger mit den kleineren Steinen anzufangen.

Die Anzahl verschiedener Zerlegungen pro Umriss ist sehr ungleich; sie schwankt zwischen 1 (beim Quadrat) und 36 (beim rechtwinkligen Trapez mit den Seiten 3, 2, 5 und $2\sqrt{2}$). Im ganzen existieren 142 Lösungen.

Die totale Rechenzeit (d.h. einschliesslich Übersetzung des ALGOL-Programms in den maschineninternen Code sowie Ausgabe der Resultate auf das Magnetband als Zwischenträger) betrug auf einer CDC 1604-A für Tangram mit Programm 2 88 Minuten. Mit Programm 3 dagegen nur noch 6 Minuten 40 Sekunden, obschon das letztere Programm allgemeiner ist. Neben einer Anzahl von kleineren und grösseren Kunstgriffen, welche zeitsparend wirken sollten, ist wohl vor allem die Vorausberechnung der Umrisse für diese drastische Reduktion verantwortlich.

Für die Wiedergabe der Lösungen wurde zunächst der normale Zeilendrucker verwendet. Die Kanten unserer Figuren lassen sich recht gut zum Beispiel durch Reihen von Sternchen darstellen, und wenn man beachtet, dass auf dem Druckgerät der Abstand von zwei Zeichen in der Zeile in grober Approximation gleich dem halben Zeilenabstand ist, erhält man auch die schrägen Geraden in brauchbarer Form. Allerdings muss durch ein ziemlich kompliziertes Unterprogramm auf Grund der Koordinaten der Kantenenden für jede Zeile berechnet werden, an welchen Stellen Sterne zu drucken sind.

Figur 7 gibt einen Ausschnitt aus dem Zeilendrucker-Output, wobei zur Illustration des Suchprozesses jedesmal, wenn ein Stein mit Erfolg gesetzt werden konnte, d.h. auch den Symmetrie-Test bestand, ein Zwischenresultat herausgegeben wurde. Zur besseren Unterscheidung wurden für den äusseren Umriss Sterne, für die gesetzten Steine der Buchstaben *X* verwendet. Wie man sieht, konnte z.B. nach der vierten und fünften Zwischenfigur der fünfte Stein (das Dreieck mit der Hypotenuse 2) nicht mehr plaziert werden. Deshalb musste in der sechsten Figur der dritte Stein (Quadrat), und in der neunten Figur sogar der zweite Stein (grosses Dreieck) an eine neue Stelle gesetzt werden.



Figur 7

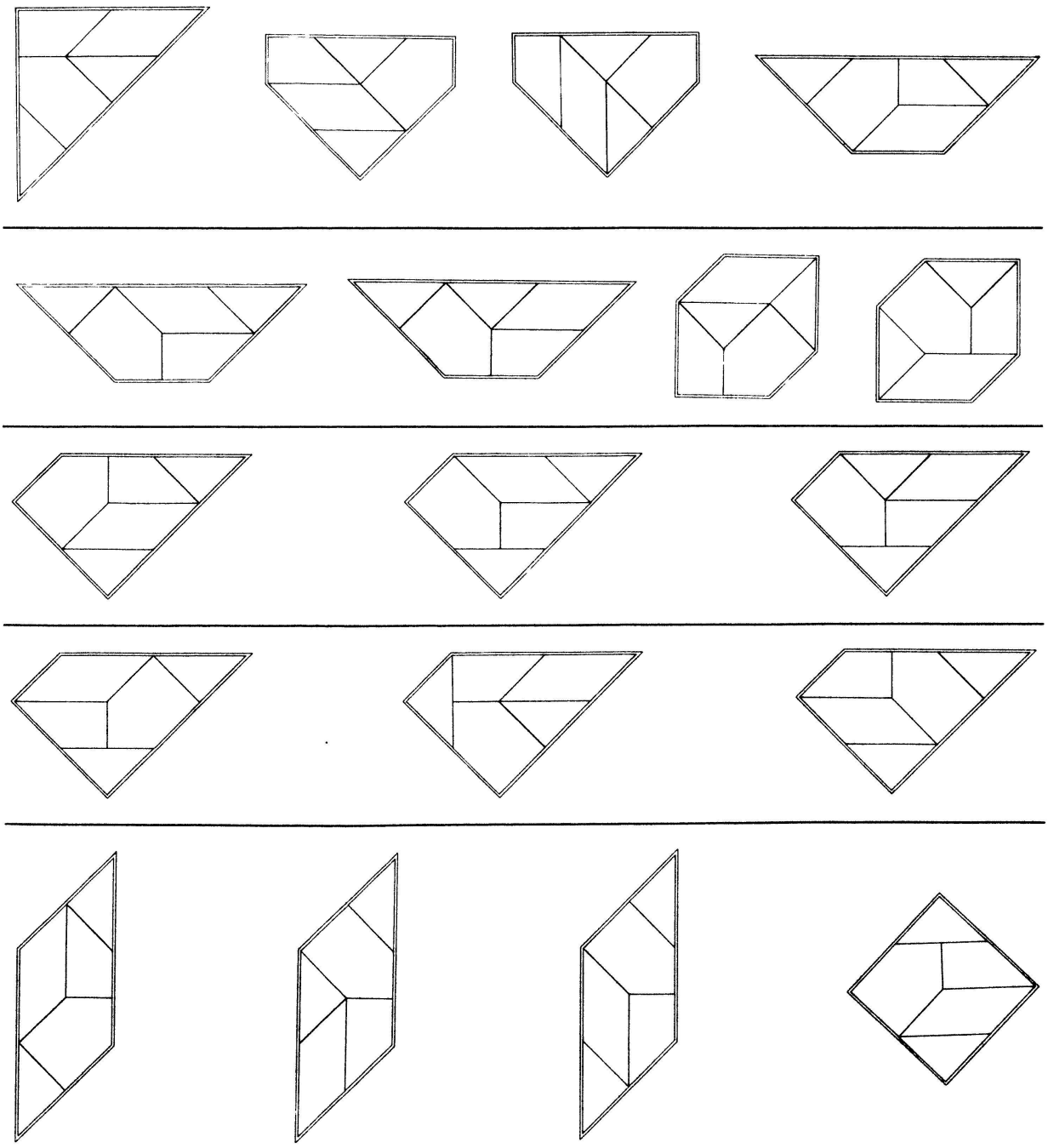
Sämtliche Zwischenresultate bis zur ersten Zerlegung eines bestimmten Umrisses im Tagram-Beispiel. Reproduktion eines Blattes, das der automatische Zeilendrucker liefert, kolonnenweise zusammengefügt.

Wesentlich schönere Figuren erhält man mit einem richtigen Zeichengerät (Plotter). In Figur 8 sind die 18 Lösungen für ein anderes, etwas einfacheres Puzzle wiedergegeben, welches übrigens, in Holz ausgeführt, als Kinderspielzeug existiert. Auch diese Figuren sind selbstverständlich, einschliesslich doppelt ausgezogenem Umriss vollautomatisch, d.h. durch das Computer-Programm gesteuert, entstanden.

8 Schliesslich möge nochmals kurz auf die am Schluss von Abschnitt 2 angeschnittenen Fragen zurückgekommen werden:

Zu 1. sind in Abschnitt 6 genügend Beispiele geliefert worden.

Was 2. betrifft, so dürfte aus den bisherigen Ausführungen hervorgegangen sein, dass in unserem Programm ziemlich extrem darauf tendiert wurde, Zeit herauszuwirtschaften, was gewiss auch Anlass zu einigen interessanten Überlegungen bot. Auf der anderen Seite muss aber doch betont werden, dass man es sich in manchen praktischen Fällen, vor allem wenn ein Programm nicht ständig wieder gebraucht wird, nicht leisten kann, soviel Arbeit in die Programmierung zu investieren wie dies beim vorliegenden Lehrbeispiel geschah, sondern dass man lieber etwas längere Rechenzeiten in Kauf nimmt. Hier muss die Erfahrung helfen, von Fall zu Fall das richtige Mass zu finden.



Figur 8

Die vollständige Lösung für ein Beispiel mit fünf Steinen (worunter zwei gleiche).
Reproduktion eines Streifens, welcher aus dem automatischen Zeichengerät (Plotter) kommt.

Die unter 3. aufgeworfene Frage der Zuverlässigkeit eines Programms (richtiges Funktionieren des Computers vorausgesetzt) darf in ihrer Bedeutung nicht unterschätzt werden. Ganz besonders peinlich wird das Problem bei Programmen wie unserem zweiten, da dort beim Ausprüfen keine Parameter variiert werden können. Hier bleibt nicht viel anderes übrig als das folgende Vorgehen: In der ganzen Testphase wird ohnehin einmal die Rechenzeit für einen Lauf des Programms auf einen kleinen Bruchteil der Zeit beschränkt, welche später für die vollständige Durchrechnung des Problems benötigt wird. Dann kommt zuerst die Ausmerzung von Schreib- und Tippfehlern (Programm auf Lochkarten!), welche zu Verstössen gegen die Grammatik der Programmsprache führen, und infolge derer das Programm überhaupt noch nicht zur Ausführung gelangt, oder welche, wenn das Programm schon läuft, groben Unsinn produzieren. Wenn dieses Stadium überwunden ist, müssen an möglichst vielen Stellen durch zusätzliche, später wieder zu eliminierende Druckbefehle, Zwischenresultate herausgegeben werden, welche in mühsamer Kleinarbeit nachzuprüfen sind. Für Programm 2 wurde noch der folgende, allerdings schwache Test durchgeführt: Völlig unabhängig vom eigentlichen Programm wurden (wie später in Programm 3) sämtliche konvexen Achtecke der Fläche 16 konstruiert und mit den Lösungsfiguren verglichen, die ja hier auf ganz andere Art entstanden waren.

Zur Illustration der angetönten Schwierigkeiten sei das folgende Detail erwähnt: Programm 2 lieferte, nachdem es vermeintlich auf Herz und Nieren geprüft und zu einem vollständigen Durchlauf frei gegeben worden war, 141 Lösungen. Als dann einige Monate später mit dem allgemeinen Programm 3 nochmals das Tangram-Beispiel gerechnet wurde, kamen zu unserer Überraschung 142 Lösungen heraus. Daraufhin wurde natürlich Programm 2 wieder vorgenommen, und eine nochmalige sorgfältige Analyse ergab, dass tatsächlich infolge eines kleinen Überlegungsfehlers, der in einer etwas komplizierten Symmetrie-Bedingung drin steckte, eine Lösung unterschlagen worden war.

Programm 3 bietet gegenüber dem vorhergehenden den Vorteil, dass es anhand von leicht überprüfbaren einfachen Fällen (nur zwei oder drei Steine) durchgetestet werden konnte. Der Verfasser glaubt, mit ziemlicher Sicherheit annehmen zu dürfen, dass es richtig funktioniert; eine absolute Sicherheit besteht jedoch auch hier nicht.

Die bisherigen Ausführungen mögen genügen, um eine Übersicht zu vermitteln. Der Verfasser ist aber gerne bereit, allfälligen Interessenten über weitere Einzelheiten betreffend Programm und Resultate Auskunft zu geben.

(Nach der Einreichung dieses Aufsatzes wurde der Verfasser auf eine frühere Arbeit hingewiesen, welche sich mit ähnlichen Fragen befasst: FU TRAING WANG and CHUAN-CHIH HSIUNG, *A Theorem on the Tangram*, Amer. Math. Monthly *49*, 596 (1942)).

Verwandte Probleme wurden auch an anderen Stellen bearbeitet; als Beispiel sei auf das Pentomino-Puzzle hingewiesen (siehe [1]). Die Pentomino-Probleme sind insofern einfacher als unsere Tangram-artigen, als dort nur im reinen Quadratnetz, d.h. ohne Schräglinien, gearbeitet wird, und dass von jedem Stein nur ein Exemplar verwendet wird, wodurch die Symmetrieschwierigkeiten weitgehend wegfallen.

Die Baumstruktur, welche ja bei unseren Betrachtungen stark im Vordergrund stand, ist selbstverständlich nicht an solche Aufgaben geometrischer Natur gebunden, sie ist vielmehr typisch für eine ganze Klasse von nichtnumerischen Problemen. Als

einziges Beispiel möge die automatische Berechnung von Stundenplänen angeführt werden (siehe [4]). Dort geht es, im Gegensatz zu unserer Puzzle-Aufgabe, nicht darum, alle Lösungen zu finden, sondern man wird im allgemeinen damit zufrieden sein, *eine* Lösung gefunden zu haben, welche mit allen Nebenbedingungen verträglich ist, und dann die Durchmusterung des Baumes abbrechen. Allenfalls können noch Optimierungsforderungen dazu kommen.

Ganz allgemein muss bei kombinatorischen Problemen unterschieden werden zwischen der Frage nach der Anzahl von Lösungen und der Aufgabe, die Lösungen effektiv zu konstruieren. Nun ist es aber so, dass auch in Fällen, da die erste der beiden Aufgaben einigermassen elementar gelöst werden kann, die zweite zum mindesten nicht ganz trivial zu sein braucht, und dass dann die Aufgabe, einen vernünftigen Algorithmus (lies: ein Computerprogramm) aufzustellen, oft sehr reizvoll ist. Eine hübsche Zusammenstellung von Problemen, die es zum Teil verdienen, auch von diesem Standpunkte aus betrachtet zu werden, findet man in [3].

P. LÄUCHLI, Zürich

LITERATURVERZEICHNIS

[1] FLETCHER, J. G., *A Program to Solve the Pentomino Problem by the Recursive Use of Macros*, Comm. ACM *8*, 621 (1965).
[2] PÓLYA, G., und SZEGÖ, G., *Aufgaben und Lehrsätze aus der Analysis*, 1. Bd., 3. Aufl., Springer-Verlag, Berlin usw. 1964.
[3] YAGLOM, A. M., and YAGLOM, I. M., *Challenging Mathematical Problems with Elementary Solutions*, Vol. 1, Übersetzung aus dem Russischen ins Englische, Holden-Day, Inc., San Francisco 1964.
[4] ZEHNDER, C. A., *Berechnung von Stundenplänen und Transportplänen*, Diss. ETH, Verlag Industrielle Organisation, Zürich 1965.

The Congruence $2^{p-1} \equiv 1 \pmod{p^2}$ and Quadratic Forms with High Density of Primes

(In memory of N. G. W. H. BEEGER, who died October 5, 1965)

There are not too many great mathematicians in the field where number theory and recreational mathematics overlap, and the most recent decennium took two away: Maurice Borisovich KRAITCHIK, who died August 19, 1957, in Brussels, 75 years old, and Nicolaas George Wijnand Henri BEEGER, who died in Amsterdam, over 80 years old. Both were friends and enriched each other and the world with their fruitful work.

In honoring N. G. W. H. BEEGER it may be permitted to digress a little from the subject. Beeger's modesty and unselfishness went so far, that as head of a commission for publishing the prime numbers of the 11th million and as the editor of this work, he didn't even mention himself in the title [2][1]), or, when he found in 1938 the quadratic form $x^2 + x - 53509$ with high density of primes, the smallest prime factor appearing in it being 61, he communicated this pearl to Luigi POLETTI of Pontremoli

[1]) Numbers in brackets refer to References, page 88.